Preface

In the past few years, many companies have been forced to reorganize their business processes by using heterogeneous technologies in order to remain competitive in a global business environment. Companies increasingly like to focus on their core expertise area and use Information Technology (IT) services to address all their peripheral needs in their business processes. In 2003, a new scientific discipline called “Services Computing,” which covers the science and technology of leveraging computing and IT to model, create, operate, and manage business processes from both theoretical and technical perspectives. In addition, Services Computing shapes the thinking of business modeling, business consulting, solution creation, service delivery, and software architecture design, development and deployment. The global nature of Services Computing leads to many opportunities and challenges and creates a new networked economic structure for supporting different business processes in these days. One of the key technological components in Services Computing is called Web services.

Current trends in Information and Communication Technology (ICT) such as Software as a Service (SaaS) and Cloud Computing have further accelerated the widespread use of Web services in business processes. In this book, a Web service is defined as an autonomous unit of application logic that provides either some business functionality or information to other applications through an Internet connection. Web services let individuals and organizations do business over the Internet using standardized protocols to facilitate application-to-application interaction. Web services offer many benefits, including platform and vendor independence, faster time to production, and convergence of disparate business functionalities. In general, there are three major entities in the Web services model: (1) A provider is the person or organization that provides an appropriate Web service for a particular business purpose; (2) A requestor is a person or organization that seeks to use a provider’s Web service to meet business requirements; and (3) A broker, or discovery agency, acts as a matchmaker between the Web services provider and requestor.

Web services are fundamentally based on a set of eXtensible Markup Language (XML) standards, such as Web Services Description Language (WSDL), Simple Object Access Protocol (SOAP), and Universal Description, Discovery and Integration (UDDI). Each service makes its functionality available through well-defined or standardized XML-format Application Programming Interface (API) on the Internet. The result of this approach is called Services Oriented Architecture (SOA). The life cycle of SOA is called a publish-find-bind model. In the publish phase, the Web services provider uses the WSDL to describe its service’s technical details. A WSDL document describes the Web service’s interface, such as which operations the Web service supports, which protocols to use, and how to pack the exchanged data. Eventually, this WSDL document will serve as a sort of contract between the Web service provider and requestor. The provider publishes the WSDL document to a Web services broker via universal description, discovery, and integration registries. UDDI is like a “yellow pages” of WSDL documents. In the
find phase, UDDI provides a standard means for organizations to describe their businesses and services and publish them so requestors can discover them online. In this scenario, the Web services broker serves as a discovery agency to help requestors find Web services that match their specific requirements. Once requestors find a Web service at the UDDI registries, they enter the bind phase, requesting the service’s corresponding WSDL document so that they can attempt to bind with the service via a SOAP message. SOAP, an XML-based messaging protocol, is independent of the underlying transport protocol (HTTP, SMTP, FTP, and so on). Service requestors use SOAP messages to invoke Web services, and Web services use SOAP messages to answer the requests. The Web service thus receives the input SOAP message from the requestor and generates an output SOAP message to the requestor.

From the architectural perspective, Web services each have a unique Uniform Resource Identifier (URI) located at a Web server on the Internet. Web services can be defined, described, and discovered using SOAP messages, which are typically on HTTP binding. On the other side, the Web services clients can be any device: a computer, tablet PC, or even a smart phone. Different systems interact with the Web service using SOAP messages, in a manner prescribed by the service description. Today, nearly all major computing companies, including Microsoft, IBM, SAP, Amazon, Sun, Oracle, and Hewlett-Packard, provide Web services tools. There are seven key properties in Web services: (1) *Loosely coupled:* Web services can run independently of each other on entirely different implementation platforms and runtime environments; (2) *Encapsulated:* The only visible part of a Web service is the public interface, such as WSDL and SOAP; (3) *Standard protocols and data formats:* Interfaces are based on a set of standards, such as XML, UDDI, WSDL, and SOAP; (4) *Invoked over an intranet or the Internet:* Web services can be executed within or outside a firewall; (5) *Components:* Web services composition can enable business-to-business transactions or connect separate enterprise systems, such as those related to workflow; (6) *Ontology:* All interacting entities must understand the functionality behind the data value computations; and (7) *Business-oriented technology:* Web services are not end-user software.

To stay competitive, companies must be agile in adapting their business processes to the ever-changing market dynamics. The adaptive business process based enterprises should look beyond the traditional enterprises and marketplaces through collaborative interactions and dynamic e-business solution bindings. The enterprise infrastructure has to provide the capability for dynamic discovery of trading partners and service providers as well as enabling federated security mechanisms, solution monitoring, and management. Thus business processes have played an important role in enabling business application integration and collaboration across multiple organisations. The integration can be categorised into two types: internal integration and external integration. Internal integration includes all the integration aspects within one enterprise. Enterprise application integration (EAI) is a typical example of internal integration. External integration covers all the possible integration patterns across multiple enterprises. The typical business process based external application integration includes business process to application integration as well as business process to business process integration.

For the external integration, traditional business-to-business applications connect trading partners through a centralised architecture. A major drawback is that setting up an additional connection with another trading partner is costly and time consuming. In contrast, the benefits of adopting Web services include faster time to production, convergence of disparate business functionalities, a significant reduction in total cost of development, and easy to deploy business applications for trading partners. In particular, Grid and Cloud infrastructures increase the need for sharing and coordinating the use of Web services for different business processes in a loosely coupled execution environment. In this context, a business process contains a set of activities which represent both business tasks and interactions between Web
services. It is believed that the major adopters of Web services include several industries that involve a set of diverse trading partners working closely together in a highly competitive market such as insurance, financial services and high technology industries. Another difference between traditional business-to-business applications and Web Services is a secure environment versus an exposed environment.

This book aims to cover major components of the lifecycle of innovation research and enabling technologies in Web services composition between the business level and IT implementation level, which includes enterprise modeling, business consulting, solution creation, services delivery, services orchestration, services optimization, services management, services marketing, services delivery and cloud computing, business process integration and management, and Web services technologies and standards. This book also aims to promote and coordinate developments in the field of business process integration and management with Web services. The chapters in this book crystallize the emerging technologies, evolution, challenges, and trends into positive efforts to focus on the most promising and innovative solutions with related issues in Web service composition. The chapters are categorized into four sections: (1) Services Composition; (2) Matchmaking and Substitution; (3) Quality of Services (QoS); and (4) Security and Privacy. It is expected that the covered topics in the chapters would further research new best practices and directions in this challenging research area. In summary, the chapters provide clear proof that Web service technologies are playing a more and more important and critical role in supporting business processes and applications.

SECTION 1 - SERVICES COMPOSITION

There is a growing need for an integrated view of Web services from different sources with the blooming of information sources and services over the Web. Automation for the assembly a coherent view of distributed heterogeneous Web services and information processing resources is a challenging and important process for inter- or intra-organizational collaboration and service provision. One of the major goals of Web services is to make easier their composition to form more complex services. In particular, a current trend in the research community is to define coordination mechanisms to execute collaborative tasks involving multiple organizations. To this purpose, many emerging languages, such as Web Service Business Process Execution Language (WS-BPEL) and Business Process Model Language (BPML) have been proposed to coordinate Web services into a workflow.

Nowadays a business process is supported by a Workflow Management System (WFMS). A workflow is a computer supported business process. WFMS is the software to support the specification, decomposition, execution, coordination, and monitoring of workflow. In general, a workflow includes many different entities, such as, activities, humans, agents, events, and flows. An event is an atomic occurrence of something interesting to the system itself or user applications. Events arise during the execution of an activity. Usually events are classified into four types: control, data, exception, and security. A flow is a directed relationship that transmits events from a source activity to a sink activity. Thus, events partition activity relationships into control-flows, data-flows, security-flows, and exception-flows. There may be constraints among flows. A workflow is a set of activities connected by flows. One can imagine that every activity starts when one or more relevant events arrive, and when the activity finishes it also generates one or more events to other dependent activity/activities. The combination of workflow technologies and Web services as a model of services composition has become more and more important in both the research community and the industry.
Despite great interests in services composition, complicated technical issues and organizational challenges remain to be solved. Thus, the management of Web services composition poses a very challenging problem. The integration of Web services into business processes is imminent. Interoperability with Web services is also an important issue in services integration. This section contains four chapters related to modelling, technical architecture, and theoretical choreographies analysis with illustrative examples in services composition shown as follows.

In chapter one, Rouached et al. present a semantic modelling approach of a process defined for a multiple Web service compositions extended from the mapping from WS-BPEL to event Calculus, and including Web service interfaces in WSDL for use in modelling between services process behaviour and choreography. For verification and validation techniques, this chapter also describes an algorithm as part of the analysis to semantically check and link partner process interactions based on automated induction-based theorem. In conclusion, this chapter gives the readers a conceptual overview of services composition modelling and analysis.

In chapter 2, Alchieri et al. present a dependable infrastructure, called WS-DependableSpace (WSDS), for cooperative stateless Web services coordination with an implementation prototype. WSDS is based on the tuple space coordination model to share data and synchronize their actions. This infrastructure shows several decoupled communication and implements several security mechanisms that allow dependable coordination even in presence of malicious components. The WSDS architecture integrates several dependability and security properties such as reliability, integrity, confidentiality, and availability in a modular way. In conclusion, this chapter gives the readers an illustrative technical architecture for supporting services composition.

In chapter 3, Xiong et al. present a model of multiple Web service interaction based on a Petri net called Composition net (C-net). In service composition, mismatches at the interface and protocol levels may render the composite service unusable. The protocol-level mismatch problem is transformed into the empty siphon problem of a C-net in this model. Further this chapter takes future deadlock states into consideration through this model, while finding the optimal solution that involves fewest interactions with a developer. In conclusion, this chapter gives the readers a mathematical model to find out the mismatch patterns at the interface and protocol level in services composition.

In the final chapter in section 1, Aktas et al. present a Grid information service architecture called Hybrid Grid Information Service (Hybrid Service) that provides unification, interaction and federation of metadata instances of grid information services with a prototype implementation and its evaluation. This chapter discusses unification, federation, interoperability, and performance aspects of the system. Based on the fundamental knowledge in services composition discussed in Chapter 1-3, this chapter concludes this topic with an illustrative example based on Grid infrastructure in services composition.

SECTION 2 - MATCHMAKING AND SUBSTITUTION

Due to the nature of SOA, the service oriented applications present a different characteristic on the issue of “availability” than traditional network applications. The application logic, defined by the SOA, requires that the applications are designed and architected in a way that makes them naturally decomposed into the components of large granules with well-circumscribed functional responsibilities. These components are delivered by loosely-coupled, autonomous and dynamically bound network-accessible software. Unlike traditional distributed applications, an application by the SOA can no longer assume a direct control over its components. This is due to the facts that the components may belong to the different owners, that they may be built and designed to run on the different platforms, and that they
can conform to the agreements of different service levels on the quality and performance. The unique characteristic of service-oriented applications posts new requirement from service oriented applications on handling availability of Web services, which forms a challenge that has not been studied in traditional fault-tolerant computing. When an application author is no longer allowed to manipulate the component services, the best way is to arrange substitute Web service to function as a replacement.

On the other side, one of the major processes in a loosely coupled Web services execution environment is matchmaking, that is, an appropriate Web service is assigned to satisfy a requestor’s requirements with or without the assistance of a service locator. Alternatively, matchmaking can also provide a ranked list of the n best candidates with respect to the requestor’s requirements. This section contains of four chapters which are related to semantic model, data structure, functional and non-functional requirements, with illustrative examples in matchmaking and substitution shown as follows.

In chapter 5, Wang et al. present environment ontology of service capabilities in terms of the domain-specific environment entities from the application scenario and the effects imposed by Web service on the entities. A hierarchical state machine is constructed for each environment entity to describe its behaviors, and the effects imposed by Web services are described as the state transitions traces of environment entities, which define the capability specification of the Web service. In conclusion, this chapter gives the readers an overview of ontology to model Web services’ capabilities.

In chapter 6, Peng et al. present a theoretical model called RSLattice based on Formal Concept Analysis (FCA) method which is applied to reveal the pairwise replaceable relationship among Web services at the operation level with an experimental evaluation of the model. Web service replaceability refers to the ability of substituting one service for another. Once RSLattice has been built for a specific service collection, searching replaceable services in the collection should be very efficient. In conclusion, this chapter gives the readers a formal model of supporting Web service replacement in a systematic approach.

In chapter 7, Bravo and Alvarado present a set of structural and functional measures to support Web service substitution with experimental tests. The process for calculating these measures is implemented using a structural-syntactical and functional filtering approach. Web service substitution occurs when, in a composite scenario, a service operation is replaced to improve the composition performance or fix a disruption caused by a failing service. In conclusion, this chapter describes a sample measure methodology to support Web service substitution in a probability approach.

In chapter 8, the final chapter of section 2, Jegadeesan and Balasubramaniam present service flavors which are a strategy for service providers to differentiate services by the non-functional aspects in a marketplace. This chapter models differentiating aspects as policies and also provides a mechanism for enforcing these policies in the middleware by a flavoring SOAP intermediary acting as a policy enforcement point. The strategy could also be used to provide services to business partners or consumers based on custom Service Level Agreements (SLAs). Based on the fundamental knowledge shown in Chapter 5-7, this chapter concludes this topic with an illustrative example of marketplace to conduct Web service’s matchmaking and substitutions and result in SLAs. A SLA is a formal contract between a Web services requestor and provider guaranteeing quantifiable issues at defined levels only through mutual concessions.

SECTION 3 – QUALITY OF SERVICES (QoS)

Quality of Services (QoS) management is critical for service-oriented enterprise architectures because services have different QoS characteristics and their interactions are dynamic and decoupled. For the
publish/subscribe style of SOA, different publishers and subscribers may have different QoS requirements in terms of performance, response time, availability, throughput, reliability, timeliness, and security. Usually, there is more than one Web services claim that they have the same or very similar capabilities to accomplish a requestor’s requirements. In many cases, the QoS may vary from Web service to Web service. In many cases, the majority of Web services providers may not concern about the level of QoS provided to their requestors. However, there exist an increasing number of concerns to maintain their popularity and reputation about the QoS. It is obvious that the QoS perceived by the requestors is thus becoming a dominant factor for the success of a Web service. In general, the principal QoS attributes of a Web service include a diverse set of service requirements such as the service availability, performance, time, efficiency, reliability, scalability, dependability and security. Matchmaking can also be based on binding support, historical performance, and QoS classifications. As Web services become more popular and complex, the need for locating Web services with specific capabilities at the service locator become more important. This section contains of four chapters which are related to mathematical model, cache data structure, and Web 2.0 with illustrative examples in QoS management shown as follows.

In chapter 9, Rosario et al. present a comprehensive framework for single or composite QoS management in business processes based on soft probabilistic contracts. This chapter also discusses contract composition on how to derive QoS contracts for an orchestration from the QoS contracts, and contract monitoring. QoS relates to performance that a Web service outperforming its contract should do well for the orchestration with SLA. In conclusion, this chapter describes a technical contract framework in an orchestration approach.

In chapter 10, Zheng and Lyu present a distributed fault tolerance strategic framework for Web services failures with a prototype. This chapter also present the optimal fault selection algorithm, which employs both the QoS performance of Web services and the requirements of service users for selecting optimal fault tolerance strategy. In conclusion, this chapter gives the reader an overview of fault tolerance methodology with an implementation.

In chapter 11, Lecue and Mehandjiev present an optimal approach to select semantic Web service compositions within a stochastic model designed to balance semantic links with QoS metrics and search method, which is formalized as a Constraint Satisfaction Problem (CSP) with multiple constraints. The overall architecture has also been tested in three realistic scenarios in use. In conclusion, this chapter gives the reader an overview of semantic methodology to support service compositions with real examples.

In chapter 12, the final chapter of section 3, Hassan et al. present a caching framework called MACE, which is specifically designed for the performance and scalability aspects of mashups. MACE is based upon a formal mashup model wherein an individual mashup is represented as a tree of operators. MACE’s design includes a dynamic mashup cache point selection scheme which maximizes the benefit of mashup caching. In conclusion, this chapter presents a set of experiments studying the effectiveness of the proposed framework in mashups.

SECTION 4 – SECURITY AND PRIVACY

Web services architectures are built on an insecure, unmonitored, and shared environment, which is open to events such as security threats. This may result in conflicts since the open architecture of Web services makes it available to many parties, who may have competing interests and goals. For example, a party’s commercial secrets may be released to another competing company via the Web services execution. As
is the case in many other applications, the information processed in Web services might be commercially sensitive so it is important to protect it from security threats such as disclosure to unauthorized parties. Since security is an essential and integral part of many business processes, Web services have to manage and execute the activities in a secure way. However, the research area of Web services security is challenging as it involves many disciplines, from authentication/encryption to access management/security policies. Security concerns and the lack of security conventions are the major barriers that prevent many business organizations from implementing or employing Web services composition.

On the other hand, privacy policies describe an organization’s data practices, what information they collect from individuals (e.g., consumers), and what (e.g., purposes) they do with it. One can imagine that information privacy is usually concerned with the confidentiality of the sensitive information. One of the most significant objectives of enforcing privacy policies is to protect Personal Identifiable Information (PII). Privacy control is usually not concerned with individual subjects. A subject releases his/her data to the custody of an enterprise while consenting to the set of purposes for which the data may be used. In business-to-business activities, information privacy is usually concerned with the confidentiality of the business information. Thus, it is required to have a privacy policy for Web services composition. The privacy policy expresses clearly and concisely what the data protection mechanisms are to achieve. The policy states also the privacy the services requestor expects the services to enforce. As a result, each information security system should enforce the privacy policy stated by the organization. In this circumstance, the information security mechanism should also be embedded with privacy-enhancing technologies.

All this evidence shows the importance of integrating privacy concepts into security mechanisms for resolving the business-to-business security and privacy concerns. However, no comprehensive solutions to the various security and privacy issues have been so far defined in a business-to-business services computing application. It is because the composite services bring a new set of security and privacy challenges to be taken into account during service composition, deployment, and execution, such as confidentiality, integrity, anonymity, authentication, authorization, and availability of composite services. As security and privacy have become an essential component for all software, several security and privacy solutions for XML data have been proposed, which may be the basis to the development of security and privacy solutions for service composition. Additionally, it is important that security and privacy solutions for service compositions are not developed on a “stand-alone” basis; rather they must be integrated into the SOA life cycle, as well as with the solutions developed to fulfill the other main requirements of service compositions. This section contains five chapters related to security and privacy in Web services composition shown as follows.

In chapter 13, She et al. present an enhanced security model to control the information flow in composite services with the concepts of delegation and pass-on with a case study from a healthcare information system. Based on these concepts, new certificates, certificate chains, delegation and pass-on policies, and how they are used to control the information flow are also discussed in this chapter. In conclusion, this chapter gives the readers an overview of the fundamental properties of a security model for Web services composition.

In chapter 14, Alam et al. present a technical framework called BA4BP which enables finer-granular attestation at the Web services level for bringing trust into business processes. This chapter also discusses the behavior of individual services in a business process to overcome the inherent weaknesses of trust management through Trust Computing Group (TCG) for the measurement of the behavior of individual services in a business process. For illustration, the framework incorporates eXtensible Access Control
Markup Language (XACML) on top of business processes. In conclusion, this chapter gives an overview of a design of security architecture in supporting trust computing.

In chapter 15, Khosravifar et al. present an incentive-based reputation model for a theoretical analysis of reputation assessment of Communities of Web services (CWSs) with an implementation. The model represents a logging and auditing mechanism in order to maintain effective reputation assessment for the communities. User agents request CWSs for services and accordingly rate their satisfactions about the received quality and community responsiveness. The strategies taken by different parties are private to individual agents, and the logging file that collects feedback is investigated by a controller agent. In conclusion, this chapter shows the importance of logging and auditing mechanism to support the security framework for services compositions.

In chapter 16, Ringelstein and Staab present a technical architecture called DIALOG for protecting information privacy with a prototype for logging and collecting logs in service-oriented and cross-organizational systems. DIALOG is based on a semantic formalism and ontology to express sticky logs. There are personal, business and legal requirements for protecting privacy and Intellectual Property Rights (IPR) and allowing customers to request information about how and by whom their data was handled. In addition to Chapter 15, this chapter further elaborates the role of logging and auditing mechanisms to protect information privacy in Web services composition.

In the final chapter of the book, chapter 17, Michlmayr et al. present an integrative technical approach into a Web service runtime environment called VRESCo that addresses service provenance. In general, provenance describes the origin and well-documented history of a given object. Several security mechanisms have been implemented to guarantee access control and integrity of service provenance information. This chapter also show the performance and applicability of the integrative technical approach for both service consumers and service providers based on some illustrative examples. Based on Chapter 13 to 16, this chapter concludes this section with an integrative technical approach to support service provenance in Web services composition.

Patrick C. K. Hung
University of Ontario Institute of Technology, Canada